WWW sits the S.A.T. - Measuring Relational Similarity on the Web

Danushka Bollegala[*1] Yutaka Matsuo[*1] Mitsuru Ishizuka[*1]

[*1]The University of Tokyo

Measuring similarity between semantic relations that hold between words is an important sub-task in various natural language processing tasks. We propose a machine learning approach to measure relational similarity between word-pairs using a web search engine. First, relations that hold between the two words in a word-pair are expressed using automatically extracted lexical lexical patterns. Second, we train a two-class support vector machine to learn the contribution of various lexical patterns towards relational similarity between two word-pairs. We use SAT word-analogy questions for training and evaluating the proposed relational similarity measure. Our preliminary experimental results report a 40% SAT score.

1. Introduction

Similarity can be broadly categorized into two types: *attributional* and *relational* [3, 12]. Attributional similarity is correspondence between attributes and relational similarity is correspondence between relations. When two words have a high degree of attributional similarity, they are called *synonymous*. When two pairs of words show a high degree of relational similarity, they are called *analogous*. For example, the two analogous word-pairs: ostrich:bird and lion:cat – both implying the relation **X is a large Y** – has a high relational similarity.

Relational similarity measures are useful for numerous tasks in natural language processing such as detecting word analogies and classifying semantic relations in noun-modifier pairs. Word analogy questions have been used as a component of Scholastic Aptitude Test (SAT) to evaluate applicants to the U.S. college system. A SAT analogy question comprises a source-pairing of concepts/terms and a choice of (usually five) possible target pairings, only one of which accurately reflects the source relationship. A typical example is shown below.

Question: Ostrich is to Bird as:

a. Cub is to Bear

b. *Lion is to Cat*

c. Ewe is to Sheep

d. Turkey is to Chicken

e. Jeep is to Truck

Here, the relation *is a large* holds between the two words in the question (e.g. Ostrich and Bird), which is also shared between the two words in the correct answer (e.g. Lion *is a large* Cat). SAT analogy questions have been used as a benchmark to evaluate relational similarity measures in previous work on relational similarity [15, 12].

Noun-modifier pairs such as *flu virus*, *storm cloud*, *expensive book*, etc are frequent in English language. In fact, WordNet contains more than 26,000 noun-modifier pairs. Natase and Szpakowicz [4] classified noun-modifiers into five classes according to the relations between the noun and the modifier. Turney [12] used a relational similarity measure to compute the similarity between noun-modifier pairs and classify them according to the semantic relations that hold between a noun and its modifier.

We proposes a method to measure the relational similarity between two given pairs of words using text-snippets returned by a web search engine. Snippets provide useful information about the relations that hold between words. For example, Google[*1] returns the snippet *...the ostrich is the largest bird in the world and can be found in South Africa...* for the conjunctive query *ostrich AND bird*. This snippet alone suggests that ostrich is a large bird. The proposed method automatically extracts lexical patterns that describe the relation implied by the two words in a word-pair and computes the relational similarity between two word-pairs using a machine learning approach.

Relational similarity is a dynamic phenomenon. In particular, relations between named entities change over time and across domains. Therefore, it is costly or even impossible to manually update language resources to reflect those changes. The proposed method does not require language resources such as taxonomies or dictionaries which makes it attractive when measuring relational similarity between named entities.

2. Related Work

The Structure-mapping theory (SMT) [1] claims that an analogy is a mapping of knowledge from one domain (base) into another (target) which conveys that a system of relations known to hold in the base also holds in the target. The target objects do not have to resemble their corresponding base objects. This structural view of analogy is based on the intuition that analogies are about relations, rather than simple features. Although this approach works best when the base and the target are rich in higher-order causal structures, it can fail when structures are missing or flat [16].

Turney et al. [14] combined 13 independent modules by considering the weighted sum of the outputs of each individual module to solve SAT analogy questions. The best performing individual module was based on Vector Space Model (VSM). In the VSM approach to measuring relational similarity [13], first a vector is created for a word-pair *X:Y* by counting the frequencies of various lexical patterns containing *X* and *Y*. In their experiments they used 128 manually created patterns such as "*X* of *Y*", "*Y* of *X*", "*X* to *Y*" and "*Y* to *X*". These patterns are then used as queries to a search

連絡先: 7-3-1, Hongo, Bunkyo-ku, Tokyo, 113-8656, Japan. danushka@mi.ci.i.u-tokyo.ac.jp

[*1] http://code.google.com/apis

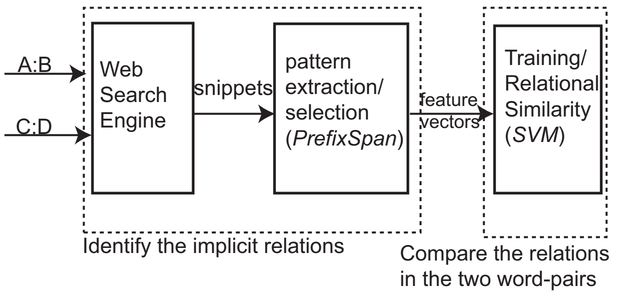
A:B
C:D
Web Search Engine
snippets
pattern extraction/ selection (PrefixSpan)
feature vectors
Training/ Relational Similarity (SVM)
Identify the implicit relations
Compare the relations in the two word-pairs

Figure 1: Outline of the proposed method.

engine and the number of hits for each query is used as elements in a vector to represent the word-pair. Finally, the relational similarity is computed as the cosine of the angle between the two vectors representing each word-pair. This VSM approach achieves a score of 47% on college-level multiple-choice SAT analogy questions. A SAT analogy question consists of a target word-pair and five choice word-pairs. The choice word-pair that has the highest relational similarity with the target word-pair in the question is selected by the system as the correct answer.

Turney [10, 12] proposes Latent Relational Analysis (LRA) by extending the VSM approach in three ways: a) lexical patterns are automatically extracted from a corpus, b) the Singular Value Decomposition (SVD) is used to smooth the frequency data, and c) synonyms are used to explore variants of the word-pairs. LRA achieves a score of 56% on SAT analogy questions. Both VSM and LRA require a large number of search engine queries to create a vector representing a word-pair. For example, with 128 patterns, VSM approach requires at least 256 queries to compute relational similarity. LRA considers synonymous variants of the given word pairs, thus requiring even more search engine queries. Despite efficient implementations, singular value decomposition of large matrices is time consuming. In fact, overall LRA takes over 8 days to process the 374 SAT analogy questions [12], which can be problematic when used in many real world NLP tasks.

Veale [15] proposed a relational similarity measure based on taxonomic similarity in WordNet. He evaluates the quality of a candidate analogy *A:B::C:D* by comparing the paths in WordNet, joining *A* to *B* and *C* to *D*. Relational similarity is defined as the similarity between the *A:B* paths and *C:D* paths. However, WordNet does not fully cover named entities such as personal names, organizations and locations, which becomes problematic when using this method to measure relational similarity between named entities.

3. Method

The proposed method is outlined in Fig. 1 and consists of two main steps: *identifying the implicit relations* between the two words in each word-pair and *comparing the relations* that exist in each word-pair. To identify the implicit relations between two words *X* and *Y*, we first query a web search engine using the phrasal query *“X*******Y”*. Here, the wildcard operator “*” would match any word or nothing. This query retrieves snippets that contain both *X* and *Y* within a window of 7 words. For example, Google returns the snippet shown in Fig.2 for the word-pair *lion:cat*. We use *PrefixSpan* (i.e., prefix-projected sequential pattern mining) [5] algorithm to extract frequent subsequences from

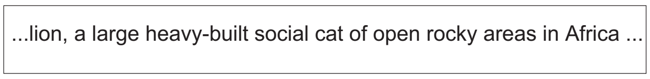
...lion, a large heavy-built social cat of open rocky areas in Africa ...

Figure 2: A snippet returned by Google for the query “lion*******cat”.

Table 1: Contingency table for a pattern v

	v	patterns other than v	Total
Freq. in snippets for question and correct answer	p_v	$P - p_v$	P
Freq. in snippets for question and incorrect answer	n_v	$N - n_v$	N

snippets that contain both *X* and *Y*. PrefixSpan extracts all word subsequences which occur more than a specified frequency in snippets. We select subsequences that contain both query words (eg. *lion* and *cat*) and replace the query words respectively with variables *X* and *Y* to construct lexical patterns. For example, some of the patterns we extract from the snippet in Fig.2 are *“X a large Y”*, *“X a large Y of”* and *“X, a large social Y”*. We select patterns that appear more than 30 times (minimum support of 30) in snippets. PrefixSpan algorithm is particularly attractive for the current task because it can efficiently extract a large number of lexical patterns.

Although automatic pattern extraction methods [7, 9] have been proposed based on dependency parsing of sentences, extracting lexical patterns from snippets using such methods is difficult because most snippets are not grammatically correct complete sentences. Moreover, the above mentioned pattern extraction method does not require part-of-speech taggers or dependency parsers, which are not available or accurate enough for all languages.

We used the SAT analogy questions dataset which was first proposed by Turney and Littman [13] as a benchmark to evaluate relational similarity measures, to extract lexical patterns. The dataset contains 2176 unique word-pairs across 374 analogy questions. For each word-pair, we searched Google and download the top 1000 snippets. From the patterns extracted by the above mentioned procedure, we select ones that occur more three times and have less than seven words. The variables *X* and *Y* in a pattern are swapped to create a reversed version of the pattern. The final set contains 9980 unique patterns. However, out of those patterns only 10% appear in both for a question and one of its choices. It is impossible to learn with such a large number of sparse patterns. Therefore, we perform a pattern selection procedure to identify those patterns that convey useful clues about implicit semantic relations.

First, for each extracted pattern v, we count the number of times where v appeared in any of the snippets for both a question and its correct answer (p_v) and in any of the snippets for both a question and any one of its incorrect answers (n_v). We then create a contingency table for each pattern v, as shown in Table 1. In Table 1, P denotes the total frequency of all patterns that occur in snippets for a question and its correct answer ($P = \sum_v p_v$) and N is the same for incorrect answers ($N = \sum_v n_v$). If a pattern occurs many times in a question and its correct answer, then such patterns are reliable indicators of latent relations between words. To evaluate the reliability of an extracted pattern as an indicator of a relation, we calculate the χ^2 [2] value for each pattern using Table 1 as,

$$\chi^2 = \frac{(P+N)(p_v(N-n_v) - n_v(P-p_v))^2}{PN(p_v+n_v)(P+N-p_v-n_v)}.$$

Patterns with χ^2 value greater than a specified threshold are used as features for training. Some of the selected patterns are shown in Table 2.

For given two pairs of words *A:B* and *C:D*, we create a feature vector. First, we record the frequency of occurrence of each selected pattern in snippets for each word-pair. We call this the *pattern frequency*. It is a local frequency count, analogous to *term frequency* in information retrieval [8]. Secondly, we multiply the two pattern frequencies of a pattern (i.e., frequency of occurrence in snippets for *A:B* and that in snippets for *C:D*) to create a feature vector for the two word-pairs.

We model the problem of computing relational similarity as a one of identifying analogous and non-analogous word-pairs, which can be solved by training a binary classifier. Using SAT analogy questions as training data, we train a two-class support vector machine (SVM) as follows. From each question in the dataset, we create a positive training instance by considering *A:B* to be the word-pair for the question and *C:D* to be the word-pair for the correct answer. Likewise, a negative training instance is created from a question word-pair and one of the incorrect answers.

The trained SVM model can then be used to compute the relational similarity between two given word-pairs *A:B* and *C:D* as follows. First, we represent the two word-pairs by a feature vector F of pattern frequency-based features. Second, we define the relational similarity $\mathrm{RelSim}(A:B, C:D)$ between the two word-pairs *A:B* and *C:D* as the posterior probability $\mathrm{Prob}(F|analogous)$ that feature vector F belongs to the analogous-pairs (positive) class,

$$\mathrm{RelSim}(A:B, C:D) = \mathrm{Prob}(F|analogous).$$

Being a large margin classifier, the output of an SVM is the distance from the decision hyper-plane. However, this is not a calibrated posterior probability. We use sigmoid functions to convert this uncalibrated distance into a calibrated posterior probability (see [6] for a detailed discussion on this topic).

4. Experiments

For the experiments in this paper we used the 374 SAT college-level multiple-choice analogy questions dataset which was first proposed by Turney et al. [14]. We compute the total score for answering SAT questions as follows,

$$\text{score} = \frac{\text{no. of correctly answered questions}}{\text{total no. of questions}}. \qquad (1)$$

Formula 1 does not penalize a system for marking incorrect answers.

Patterns with the highest linear kernel weights are shown in Table 2 alongside their χ^2 values. The weight of a feature in the linear kernel can be considered as a rough estimate of the influence it imparts on the final SVM output. Patterns shown in Table 2 express various semantic relations that can be observed in SAT analogy questions.

Table 2: Patterns with the highest SVM linear kernel weights

pattern	χ^2	SVM weight
and Y and X	0.8927	0.0105
Y X small	0.0795	0.0090
X in Y	0.0232	0.0087
use Y to X	0.5059	0.0082
from the Y X	0.3697	0.0079
to that Y X	0.1310	0.0077
or X Y	0.0751	0.0074
X and other Y	1.0675	0.0072
a Y or X	0.0884	0.0068
that Y on X	0.0690	0.0067

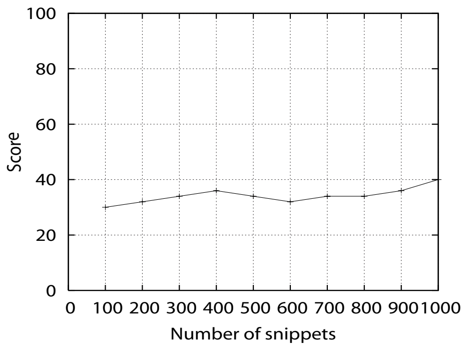

Figure 3: Performance with the number of snippets

Figure 3 plots the variation of SAT score with the number of snippets used for extracting patterns. From Fig.3 it is apparent that overall the score improves with the number of snippets used for extracting patterns. The probability of finding better patterns increases with the number of snippets processed. That fact enables us to represent word-pairs with a rich feature vector, resulting in better performance. We believe that the drop of SAT score around 600 snippets is a result of irrelevant search results and/or improper ranking by the search engine.

Table 3 summarizes various relational similarity measures proposed in previous work. All algorithms in Table 3 are evaluated on the same SAT analogy questions. Score is computed by Formula 1. Because SAT questions contain 5 choices, a random guessing algorithm would obtain a score of 0.2 (lower bound). The score reported by average senior high-school student is about 0.570 [13] (upper bound). We performed 5-fold cross validation on SAT questions to evaluate the performance of the proposed method. The first 13 (rows 1-13) algorithms were proposed by Turney et al. [14], in which they combined these modules using a weight optimization method. For given two word-pairs, the phrase vector (row 1) algorithm creates a vector of manually cre-

Table 3: Comparison with previous relational similarity measures.

	Algorithm	score		Algorithm	score
1	Phrase Vectors	0.382	11	Holonym:member	0.200
2	Thesaurus Paths	0.250	12	Similarity:dict	0.180
3	Synonym	0.207	13	Similarity:wordsmyth	0.294
4	Antonym	0.240	14	Combined [14]	0.450
5	Hypernym	0.227	15	Proposed (SVM)	0.401
6	Hyponym	0.249	16	WordNet [15]	0.428
7	Meronym:substance	0.200	17	VSM [13]	0.471
8	Meronym:part	0.208	18	Pertinence [11]	0.535
9	Meronym:member	0.200	19	LRA [10]	0.561
10	Holonym:substance	0.200	20	Human	0.570

ated pattern-frequencies for each word-pair and compute the cosine of the angle between the vectors. Algorithms in rows 2-11 use WordNet to compute various relational similarity measures based on different semantic relations defined in WordNet. **Similarity:dict** (row 12) and **Similarity:wordsmith** (row 13) respectively use `Dictionary.com` and `Wordsmyth.net` to find the definitions of words in word-pairs and compute the relational similarity as the overlap of words in the definitions. The proposed method outperforms all those 13 individual modules reporting a score of 0.401, which is comparable with Veale's [15] WordNet-based relational similarity measure. However, the fact that a combination (row 14) of 13 independent modules can significantly outperform all its components suggests that the proposed method could potentially complement the WordNet-based measures in a hybrid approach to leverage a more robust relational similarity measure.

Although LRA (row 19 in Table 3) reports the highest SAT score of 0.561 it takes over 8 days to process the 374 SAT analogy questions [12]. On the other hand the proposed method requires less than 6 hours using a similar hardware environment[*2]. The gain in speed is mainly attributable to the lesser number of web queries required by the proposed method. To compute the relational similarity between two word-pairs *A:B* and *C:D* using LRA, we first search in a dictionary for synonyms for each word. Then the original words are replaced by their synonyms to create alternative pairs. Each word-pair is represented by a vector of pattern-frequencies using a set of automatically created 4000 lexical patterns. Pattern frequencies are obtained by searching for the pattern in a web search engine. For example, to create a vector for a word-pair with three alternatives, LRA requires 12000 (4000×3) queries. On the other hand, the proposed method first downloads snippets for each word-pair and then searches for patterns only in the downloaded snippets. Because multiple snippets can be retrieved by issuing a single query, the proposed method requires only one search query to compute a pattern-frequency vector for a word-pair. Processing snippets is also efficient as it obviates the trouble of downloading web pages, which might be time consuming depending on the size of the pages. Moreover, LRA is based on singular value decomposition (SVD), which requires time consuming complex matrix computations.

5. Conclusion

We proposed a method to measure the similarity between semantic relations that are implied by pairs of words. To represent the various semantic relations that hold between two words, we proposed a pattern extraction algorithm using a web search engine. Our preliminary experimental results report a SAT score of 40%, which is comparable with WordNet-based approaches. In future, we intend to explore the possibilities of integrating the proposed method with WordNet-based approaches.

*2 we used a desktop computer with a 2.4 GHz Pentium4 processor and 2GB of RAM

References

[1] B. Falkenhainer, K.D. Forbus, and D. Gentner. Structure mapping engine: Algorithm and examples. *Artificial Intelligence*, 41:1–63, 1989.

[2] C. D. Manning and H. Schütze. *Foundations of Statistical Natural Language Processing*. The MIT Press, Cambridge, Massachusetts, 2002.

[3] D.L. Medin, R.L. Goldstone, and D. Genter. Similarity involving attributes and relations: Judgments of similarity and difference are not inverse. *Psychological Sciences*, 1(1):64–69, 1990.

[4] V. Natase and S. Szpakowicz. Exploring noun-modifier semantic relations. In *Proc. of fifth int'l workshop on computational semantics (IWCS-5)*, pages 285–301, 2003.

[5] J. Pei, J. Han, B. Mortazavi-Asi, J. Wang, H. Pinto, Q. Chen, U. Dayal, and M. Hsu. Mining sequential patterns by pattern-growth: the prefixspan approach. *IEEE Transactions on Knowledge and Data Engineering*, 16(11):1424–1440, 2004.

[6] J. Platt. Probabilistic outputs for support vector machines and comparison to regularized likelihood methods. *Advances in Large Margin Classifiers*, pages 61–74, 2000.

[7] D. Ravichandran and E. Hovy. Learning surface text patterns for a question answering system. In *Proc. of ACL '02*, pages 41–47, 2001.

[8] G. Salton and C. Buckley. *Introduction to Modern Information Retreival*. McGraw-Hill Book Company, 1983.

[9] R. Snow, D. Jurafsky, and A.Y. Ng. Learning syntactic patterns for automatic hypernym discovery. In *Proc. of Advances in Neural Information Processing Systems (NIPS) 17*, pages 1297–1304, 2005.

[10] P.D. Turney. Measuring semantic similarity by latent relational analysis. In *Proc. of IJCAI'05*, pages 1136–1141, 2005.

[11] P.D. Turney. Expressing implicit semantic relations without supervision. In *Proc. of Coling/ACL'06*, pages 313–320, 2006.

[12] P.D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

[13] P.D. Turney and M.L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60:251–278, 2005.

[14] P.D. Turney, M.L. Littman, J. Bigham, and V. Shnayder. Combining independent modules to solve multiple-choice synonym and analogy problems. In *Proc. of RANLP'03*, pages 482–486, 2003.

[15] T. Veale. Wordnet sits the sat: A knowledge-based approach to lexical analogy. In *Proc. of 16th European Conference on Artificial Intelligence (ECAI'04)*, pages 606–612, 2004.

[16] T. Veale and M. T. Keane. The competence of structure mapping on hard analogies. In *Proc. of IJCAI'03*, 2003.